CHAPTER 24. AUTOMATING SYSTEM TASKS

You can configure Red Hat Enterprise Linux to automatically run tasks, also known as jobs:

- regularly at specified time using cron, see Section 24.1, "Scheduling a Recurring Job Using Cron"
- asynchronously at certain days using anacron, see Section 24.2, "Scheduling a Recurring Asynchronous Job Using Anacron"
- once at a specific time using at, see Section 24.3, "Scheduling a Job to Run at a Specific Time Using at"
- once when system load average drops to a specified value using batch, see Section 24.4, "Scheduling a Job to Run on System Load Drop Using batch"
- once on the next boot, see Section 24.5, "Scheduling a Job to Run on Next Boot Using a systemd Unit File"

This chapter describes how to perform these tasks.

24.1. SCHEDULING A RECURRING JOB USING CRON

Cron is a service that enables you to schedule running a task, often called a job, at regular times. A **cron** job is only executedif the system is runningon the scheduled time. For scheduling jobs that can postpone their execution to when the system boots up, so a job is not "lost" if the system is not running, see Section 24.3, "Scheduling a Job to Run at a Specific Time Using at."

Users specify cron jobs in cron table files, also called**crontab** files. These files are then read by the **crond** service, which executes the jobs.

24.1.1. Prerequisites for Cron Jobs

Before scheduling a **cron** job:

1. Install the cronie package:

```
~]# yum install cronie
```

2. The **crond** service is enabled - made to start automatically at boot time - upon installation. If you disabled the service, enable it:

```
~]# systemctl enable crond.service
```

3. Start the **crond** service for the current session:

```
~]# systemctl start crond.service
```

4. (optional) Configure cron. For example, you can change:
   - shell to be used when executing jobs
   - the **PATH** environment variable

- *mail addressee if a job sends emails.*
  *See the crontab(5) manual page for information on configuring**cron**.*

24.1.2. Scheduling a Cron Job

Scheduling a Job as root User
*The **root** user uses the cron table in **/etc/crontab**, or, preferably, creates a cron table file in **/etc/cron.d/**. Use this procedure to schedule a job as **root**:*

1. *Choose:*
   - *in which minutes of an hour to execute the job. For example, use **0,10,20,30,40,50** or **0/10** to specify every 10 minutes of an hour.*
   - *in which hours of a day to execute the job. For example, use **17-20** to specify time from 17:00 to 20:59.*
   - *in which days of a month to execute the job. For example, use **15** to specify 15th day of a month.*
   - *in which months of a year to execute the job. For example, use **Jun,Jul,Aug** or **6,7,8** to specify the summer months of the year.*
   - *in which days of the week to execute the job. For example, use * for the job to execute independently of the day of week.*
     Combine the chosen values into the time specification. The above example values result into this specification:

     ***0,10,20,30,40,50 17-20 15 Jun,Jul,Aug ***
2. *Specify the user. The job will execute as if run by this user. For example, use **root**.*
3. *Specify the command to execute. For example, use **/usr/local/bin/my-script.sh***
4. *Put the above specifications into a single line:*

   0,10,20,30,40,50 17-20 15 Jun,Jul,Aug * root /usr/local/bin/my-script.sh
5. *Add the resulting line to **/etc/crontab**, or, preferably, create a cron table file in **/etc/cron.d/** and add the line there.*

The job will now run as scheduled.

*For full reference on how to specify a job, see the crontab(5) manual page. For basic information, see the beginning of the **/etc/crontab** file:*

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root

# For details see man 4 crontabs

# Example of job definition:
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
```

```
# | | | .------- month (1 - 12) OR jan,feb,mar,apr ...
# | | | | .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# | | | | |
# * * * * * user-name command to be executed
```

Scheduling a Job as Non-root User
Non-root users can use the crontab utility to configure cron jobs. The jobs will run as if executed by that user.

To create a cron job as a specific user:

1. *From the user's shell, run:*

   ```
   [bob@localhost ~]$ crontab -e
   ```

   *This will start editing of the user's own **crontab** file using the editor specified by the **VISUAL** or **EDITOR** environment variable.*

2. *Specify the job in the same way as in ???TITLE???, but leave out the field with user name. For example, instead of adding*

   ```
   0,10,20,30,40,50 17-20 15 Jun,Jul,Aug * bob /home/bob/bin/script.sh
   ```

   add:

   ```
   0,10,20,30,40,50 17-20 15 Jun,Jul,Aug * /home/bob/bin/script.sh
   ```

3. *Save the file and exit the editor.*

4. *(optional) To verify the new job, list the contents of the current user's crontab file by running:*

   ```
   [bob@localhost ~]$ crontab -l
   @daily /home/bob/bin/script.sh
   ```

Scheduling Hourly, Daily, Weekly, and Monthly Jobs
To schedule an hourly, daily, weekly, or monthly job:

1. *Put the actions you want your job to execute into a shell script.*

2. *Put the shell script into one of the following directories:*

   - */etc/cron.hourly/*
   - */etc/cron.daily/*
   - */etc/cron.weekly/*
   - */etc/cron.monthly/*

*From now, your script will be executed – the **crond** service automatically executes any scripts present in **/etc/cron.hourly**, **/etc/cron.daily**, **/etc/cron.weekly**, and **/etc/cron.monthly** directories at their corresponding times.*

24.2. SCHEDULING A RECURRING ASYNCHRONOUS JOB USING ANACRON

Anacron, like **cron**, is a service that enables you to schedule running a task, often called a job, at regular times. However, **anacron** differs from **cron** in two ways:

- If the system is not running at the scheduled time, an **anacron** job is postponed until the system is running;
- An **anacron** job can run once per day at most.

Users specify anacron jobs in anacron table files, also called **anacrontab** files. These files are then read by the **crond** service, which executes the jobs.

24.2.1. Prerequisites for Anacrob Jobs

Before scheduling an **anacron** job:

1. Verify that you have the cronie-anacron package installed:

   ```
   ~]# rpm -q cronie-anacron
   ```

   The cronie-anacron is likely to be installed already, because it is a sub-package of the cronie package. If it is not installed, use this command:

   ```
   ~]# yum install cronie-anacron
   ```

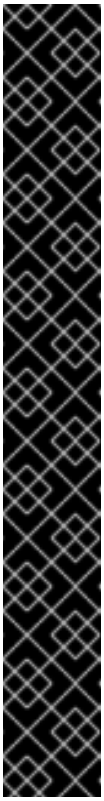
2. The **crond** service is enabled - made to start automatically at boot time - upon installation. If you disabled the service, enable it:

   ```
   ~]# systemctl enable crond.service
   ```

3. Start the **crond** service for the current session:

   ```
   ~]# systemctl start crond.service
   ```

4. (optional) Configure anacron. For example, you can change:
   - shell to be used when executing jobs
   - the **PATH** environment variable
   - mail addressee if a job sends emails.
     See the anacrontab(5) manual page for information on configuring **anacron**.



IMPORTANT

By default, the anacron configuration includes a condition that prevents it from running if the computer is not plugged in. This setting ensures that the battery is not drained by running anacron jobs.

*If you want to allow anacron to run even if the computer runs on battery power, open the **/etc/cron.hourly/0anacron** file and comment out the following part:*

```
# Do not run jobs when on battery power
online=1
for psupply in AC ADP0 ; do
  sysfile="/sys/class/power_supply/$psupply/online"

  if [ -f $sysfile ] ; then
    if [ `cat $sysfile 2>/dev/null`x = 1x ]; then
      online=1
      break
    else
      online=0
    fi
  fi
done
```

24.2.2. Scheduling an Anacron Job

Scheduling an anacron Job as root User
*The **root** user uses the anacron table in **/etc/anacrontab**. Use the following procedure to schedule a job as **root**.*

*Scheduling an anacron Job as **root** User*

1. *Choose:*
   - *Frequency of executing the job. For example, use **1** to specify every day or **3** to specify once in 3 days.*
   - *The delay of executing the job. For example, use **0** to specify no delay or **60** to specify 1 hour of delay.*
   - *The job identifier, which will be used for logging. For example, use **my.anacron.job** to log the job with the **my.anacron.job** string.*
   - *The command to execute. For example, use **/usr/local/bin/my-script.sh***
     Combine the chosen values into the job specification. Here is an example specification:

     ```
     3 60 cron.daily /usr/local/bin/my-script.sh
     ```

2. *Add the resulting line to **/etc/anacrontab**.*

The job will now run as scheduled.

*For simple job examples, see the **/etc/anacrontab** file. For full reference on how to specify a job, see the anacrontab(5) manual page.*

Scheduling Hourly, Daily, Weekly, and Monthly Jobs
You can schedule daily, weekly, and monthly jobs with anacron. See the section called "Scheduling Hourly, Daily, Weekly, and Monthly Jobs".

24.3. SCHEDULING A JOB TO RUN AT A SPECIFIC TIME USING AT

*To schedule a one-time task, also called a job, to run once at a specific time, use the **at** utility.*

*Users specify **at** jobs using the at utility. The jobs are then executed by the **atd** service.*

24.3.1. Prerequisites for At Jobs

*Before scheduling an **at** job:*

1. *Install the at package:*

```
~]# yum install at
```

2. *The **atd** service is enabled - made to start automatically at boot time - upon installation. If you disabled the service, enable it:*

```
~]# systemctl enable atd.service
```

3. *Start the **atd** service for the current session:*

```
~]# systemctl start atd.service
```

24.3.2. Scheduling an At Job

1. *A job is always run by some user. Log in as the desired user and run:*

```
~]# at time
```

Replace time with the time specification.

*For details on specifying time, see the at(1) manual page and the **/usr/share/doc/at/timespec** file.*

Example 24.1. Specifying Time for At

To execute the job at 15:00, run:

```
~]# at 15:00
```

If the specified time has passed, the job is executed at the same time the next day.

To execute the job on August 20 2017, run:

```
~]# at August 20 2017
```

or

```
~]# at 082017
```

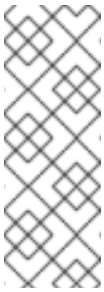
To execute the job 5 days from now, run:

```
~]# now + 5 days
```

2. At the displayed **at>** prompt, enter the command to execute and press Enter:

```
~]# at 15:00
at> sh /usr/local/bin/my-script.sh
at>
```

Repeat this step for every command you want to execute.



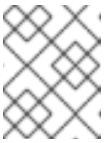
NOTE

The **at>** prompt shows which shell it will use:

```
warning: commands will be executed using /bin/sh
```

The at utility uses the shell set in user's SHELL environment variable, or the user's login shell, or **/bin/sh**, whichever is found first.

3. Press Ctrl+D on an empty line to finish specifying the job.



NOTE

If the set of commands or the script tries to display information to standard output, the output is emailed to the user.

Viewing Pending Jobs
To view the list of pending jobs, use the **atq** command:

```
~]# atq
26   Thu Feb 23 15:00:00 2017 a root
28   Thu Feb 24 17:30:00 2017 a root
```

Each job is listed on a separate line in the following format:

```
job_number scheduled_date scheduled_hour job_class user_name
```

The **job_queue** column specifies whether a job is an **at** or a **batch** job. **a** stands for **at**, **b** stands for **batch**.

Non-root users only see their own jobs. The root user sees jobs for all users.

Deleting a Scheduled Job
To delete a scheduled job:

1. List pending jobs with the **atq** command:

```
~]# atq
26    Thu Feb 23 15:00:00 2017 a root
28    Thu Feb 24 17:30:00 2017 a root
```

2. Find the job you want to delete by its scheduled time and the user.

3. Run the **atrm** command, specifying the job by its number:

```
~]# atrm 26
```

24.3.2.1. Controlling Access to At and Batch

You can restrict access to the **at** and **batch** commands for specific users. To do this, put user names into **/etc/at.allow** or **/etc/at.deny** according to these rules:

- Both access control files use the same format: one user name on each line.
- No white space is permitted in either file.
- If the **at.allow** file exists, only users listed in the file are allowed to use **at** or **batch**, and the **at.deny** file is ignored.
- If **at.allow** does not exist, users listed in **at.deny** are not allowed to use **at** or **batch**.
- The **root** user is not affected by the access control files and can always execute the **at** and **batch** commands.

The **at** daemon (**atd**) does not have to be restarted if the access control files are modified. The access control files are read each time a user tries to execute the **at** or **batch** commands.

24.4. SCHEDULING A JOB TO RUN ON SYSTEM LOAD DROP USING BATCH

To schedule a one-time task, also called a job, to run when the system load average drops below the specified value, use the **batch** utility. This can be useful for performing resource-demanding tasks or for preventing the system from being idle.

Users specify **batch** jobs using the batch utility. The jobs are then executed by the **atd** service.

24.4.1. Prerequisites for Batch Jobs

The **batch** utility is provided in the **at** package, and **batch** jobs are managed by the **atd** service. Hence, the prerequisites for **batch** jobs are the same as for **at** jobs. See Section 24.3.1, "Prerequisites for At Jobs".

24.4.2. Scheduling a Batch Job

1. A job is always run by some user. Log in as the desired user and run:

```
~]# batch
```

2. At the displayed **at>** prompt, enter the command to execute and press Enter:

```
~]# batch
at> sh /usr/local/bin/my-script.sh
```

Repeat this step for every command you want to execute.

NOTE

The **at>** prompt shows which shell it will use:

```
warning: commands will be executed using /bin/sh
```

The batch utility uses the shell set in user's SHELL environment variable, or the user's login shell, or **/bin/sh**, whichever is found first.

3. Press Ctrl+D on an empty line to finish specifying the job.

NOTE

If the set of commands or the script tries to display information to standard output, the output is emailed to the user.

Changing the Default System Load Average Limit
By default, **batch** jobs start when system load average drops below 0.8. This setting is kept in the **atq** service. To change the system load limit:

1. To the **/etc/sysconfig/atd** file, add this line:

```
OPTS='-l x'
```

Substitute x with the new load average. For example:

```
OPTS='-l 0.5'
```

2. Restart the **atq** service:

```
# systemctl restart atq
```

Viewing Pending Jobs
To view the list of pending jobs, use the **atq** command. See the section called "Viewing Pending Jobs".

Deleting a Scheduled Job
To delete a scheduled job, use the **atrm** command. See the section called "Deleting a Scheduled Job".

Controlling Access to Batch
You can also restrict the usage of the **batch** utility. This is done for the **batch** and **at** utilities together. See Section 24.3.2.1, "Controlling Access to At and Batch".

24.5. SCHEDULING A JOB TO RUN ON NEXT BOOT USING A SYSTEMD UNIT FILE

The cron, anacron, at, and batch utilities allow scheduling jobs for specific times or for when system workload reaches a certain level. It is also possible to create a job that will run during the next system boot. This is done by creating a ***systemd*** *unit file that specifies the script to run and its dependencies.*

To configure a script to run on the next boot:

1. *Create the* ***systemd*** *unit file that specifies at which stage of the boot process to run the script. This example shows a unit file with a reasonable set of* ***Wants=*** *and* ***After=*** *dependencies:*

```
~]# cat /etc/systemd/system/one-time.service
[Unit]
# The script needs to execute after:
# network interfaces are configured
Wants=network-online.target
After=network-online.target
# all remote filesystems (NFS/_netdev) are mounted
After=remote-fs.target
# name (DNS) and user resolution from remote databases (AD/LDAP) are available
After=nss-user-lookup.target nss-lookup.target
# the system clock has synchronized
After=time-sync.target

[Service]
Type=oneshot
ExecStart=/usr/local/bin/foobar.sh

[Install]
WantedBy=multi-user.target
```

   If you use this example:

   - *substitute* ***/usr/local/bin/foobar.sh*** *with the name of your script*
   - *modify the set of* ***After=*** *entries if necessary*
     For information on specifying the stage of boot, see *Section 10.6, "Creating and Modifying systemd Unit Files"**.*

2. *If you want the* ***systemd*** *service to stay active after executing the script, add the* ***RemainAfterExit=yes*** *line to the* ***[Service]*** *section:*

```
[Service]
Type=oneshot
RemainAfterExit=yes
ExecStart=/usr/local/bin/foobar.sh
```

3. *Reload the* ***systemd*** *daemon:*

```
~]# systemctl daemon-reload
```

4. *Enable the* ***systemd*** *service:*

```
~]# systemctl enable one-time.service
```

5. *Create the script to execute:*

```
~]# cat /usr/local/bin/foobar.sh
#!/bin/bash

touch /root/test_file
```

6. If you want the script to run during the next boot only, and not on every boot, add a line that disables the **systemd** unit:

```
#!/bin/bash

touch /root/test_file
systemctl disable one-time.service
```

7. Make the script executable:

```
~]# chmod +x /usr/local/bin/foobar.sh
```

24.6. ADDITIONAL RESOURCES

For more information on automating system tasks on Red Hat Enterprise Linux, see the resources listed below.

Installed Documentation

- cron - The manual page for the**crond** daemon documents how **crond** works and how to change its behavior.
- crontab - The manual page for the**crontab** utility provides a complete list of supported options.
- crontab(5) - This section of the manual page for the**crontab** utility documents the format of **crontab** files.